

Manual de Proyecto

Presentado por:
Jorge Molina Guerrero

Proyecto SDHOSTING

ÍNDICE

- 01.** Descripción General
- 02.** Instalación y Ejecución
 - 2.1 Requisitos Previos
 - 2.2 Componentes del Proyecto
 - 2.3. Pasos de Despliegue
- 03.** Tecnologías Utilizadas
- 04.** Capturas de Pantalla
- 05.** Mantenimiento y Buenas Prácticas
- 06.** Arquitectura y Diagramas

Proyecto a cargo de jorge Molina

Descripción General

Este proyecto integra varias herramientas para automatizar la provisión de infraestructuras y servicios de hosting:

- ☒ Hestia CP: Panel de control de hosting, gestiona dominios, cuentas FTP, DNS y correo.
- ☒ WHMCS: Sistema de facturación y automatización de clientes.
- ☒ n8n: Motor de automatización de flujos de trabajo, usado para orquestar comandos y notificaciones.
- Terraform: Para aprovisionar VPS.

El flujo principal: un cliente adquiere un servicio en WHMCS → webhook dispara un flujo en n8n → n8n ejecuta comandos de Hestia CLI y Terraform → configura DNS, web → envía credenciales por correo.

Instalación y Ejecución

Requisitos Previos

- ☒ Sistema Operativo: Debian 11/12 o Ubuntu LTS.
- ☒ Acceso root o usuario con sudo.
- ☒ Certificados SSL válidos para dominios.
- ☒ Docker (Para n8n y Terraform).

Componentes del Proyecto

1. Hestia CP: instalado en servidor principal.
2. WHMCS: desplegado en HestiaCP.
3. n8n: Controlador de flujo.
4. Terraform: scripts en terraform/ para infraestructura.
5. Scripts: comandos CLI en scripts/ ejecutados por n8n.

Pasos de Despliegue

1. Instalar Hestia CP:
2.

```
curl -O
https://raw.githubusercontent.com/hestiacp/hestiacp/release/install/hst-install.sh
```

Iniciar instalación

```
bash hst-install.sh --multiphp yes --apache yes --nginx yes --
phpfpm yes
```

3. Configurar dominios y SSL:

Para poder acceder de forma segura a nuestro dominio creado.

```
v-add-domain whmcs whmcs.sdhosting.dev  
v-add-web-domain-ssl whmcs whmcs.sdhosting.dev
```

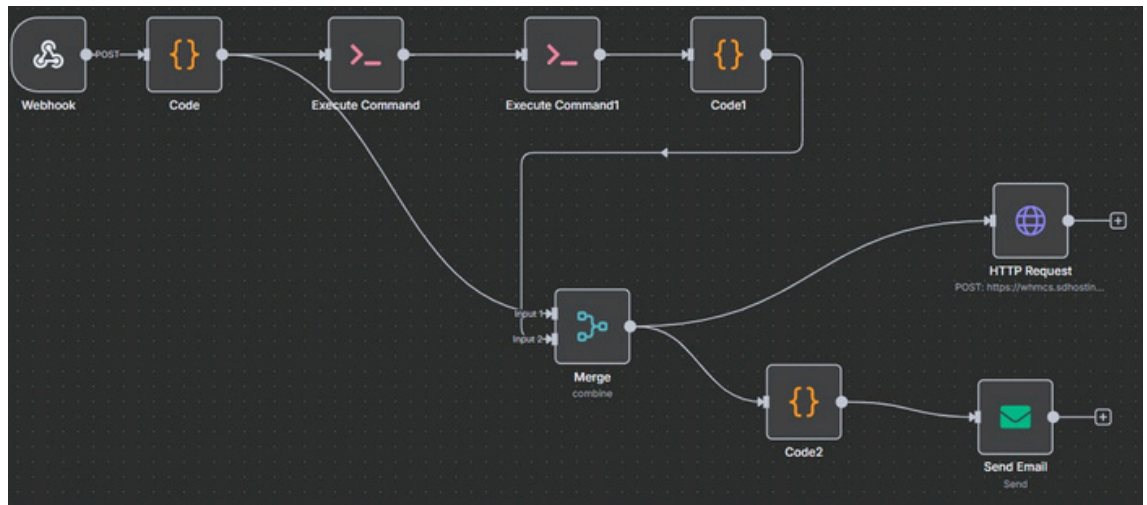
4. Instalar WHMCS:

- o Copiar archivos a /etc/whmcs/whmcs_storage, asignar permisos.
- o Crear base de datos y usuario MySQL (Se crea automáticamente).
- o Ejecutar instalador web.

5. Desplegar n8n y Terraform con Docker:

```
Version: '3.8'  
  
services:  
  terraform:  
    build:  
      context: .  
      dockerfile: Dockerfile  
    container_name: terraform-container  
    network_mode: "host"  
    dns:  
      - 1.1.1.1  
    working_dir: /workspace  
    volumes:  
      - ./config:/workspace  
    entrypoint: ["/bin/sh"]  
    tty: true  
  
  n8n:  
    build:  
      context: .  
      dockerfile: Dockerfile.n8n  
    container_name: n8n  
    ports:  
      - "5678:5678"  
    environment:  
      - N8N_BASIC_AUTH_ACTIVE=true  
      - N8N_BASIC_AUTH_USER=admin  
      - N8N_BASIC_AUTH_PASSWORD=admin123  
      - N8N_SECURE_COOKIE=false  
      - N8N_HOST=vpnsd.ddns.net  
      - N8N_PORT=5678  
      - N8N_PROTOCOL=http  
    volumes:  
      - ./n8n_data:/home/node/.n8n  
      - ./config:/workspace  
      - /var/run/docker.sock:/var/run/docker.sock  
    depends_on:  
      - terraform  
    restart: unless-stopped
```

6. flujo de n8n:

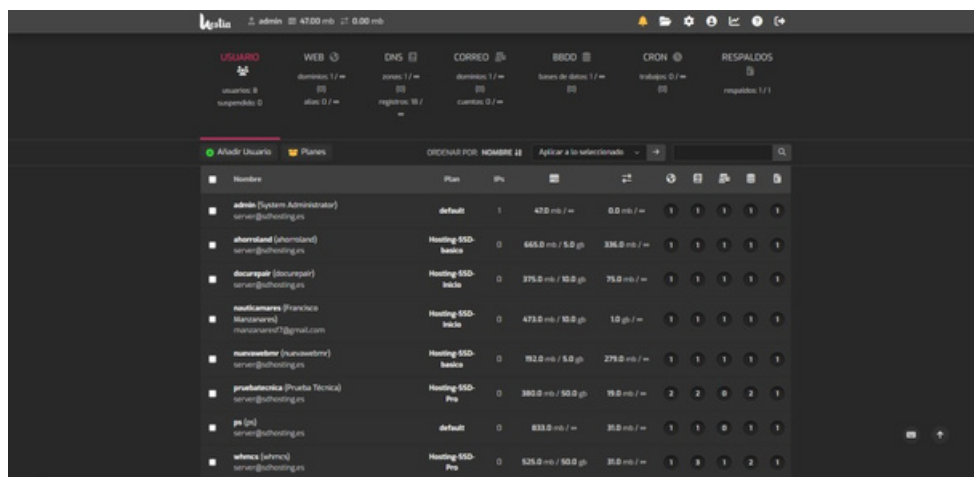


Tecnologías Utilizadas

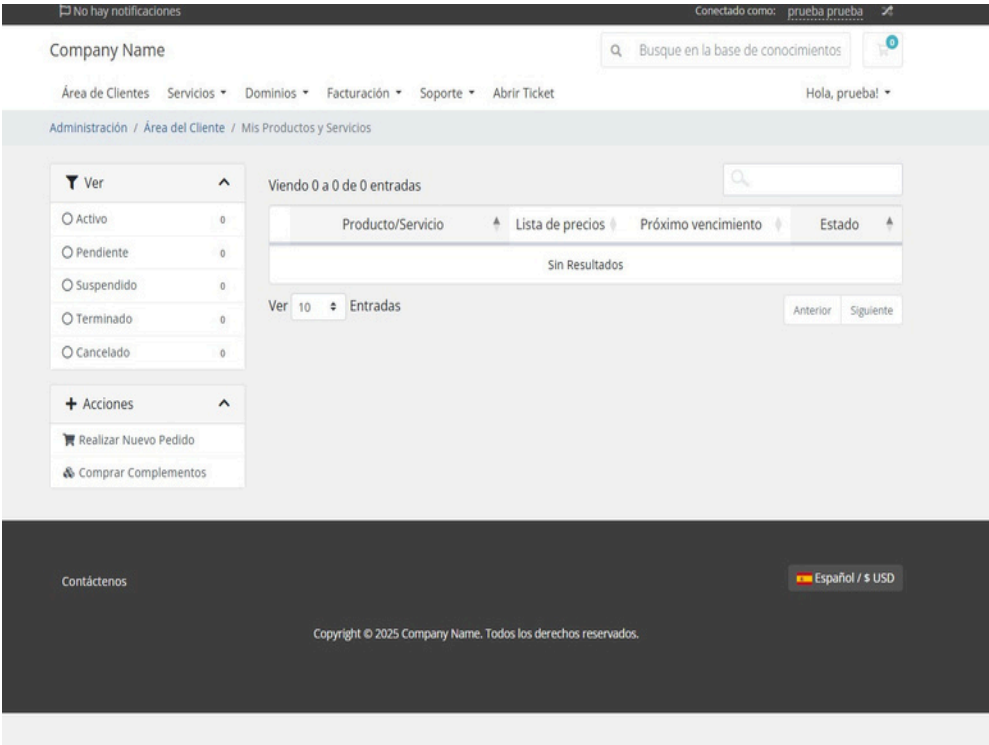
- ❑ Hestia CP: panel hosting.
- ❑ WHMCS: facturación y soporte.
- ❑ n8n: automatización de workflows.
- ❑ Terraform: Automatización en el despliegue de N8N.
- ❑ Docker: contenedores de n8n y terraform.
- ❑ MySQL/MariaDB: bases de datos.
- ❑ NGINX + Apache + PHP-FPM.
- ❑ Dovecot: correo IMAP/SMTP.
- ❑ Let's Encrypt: certificados SSL.

Capturas de Pantalla

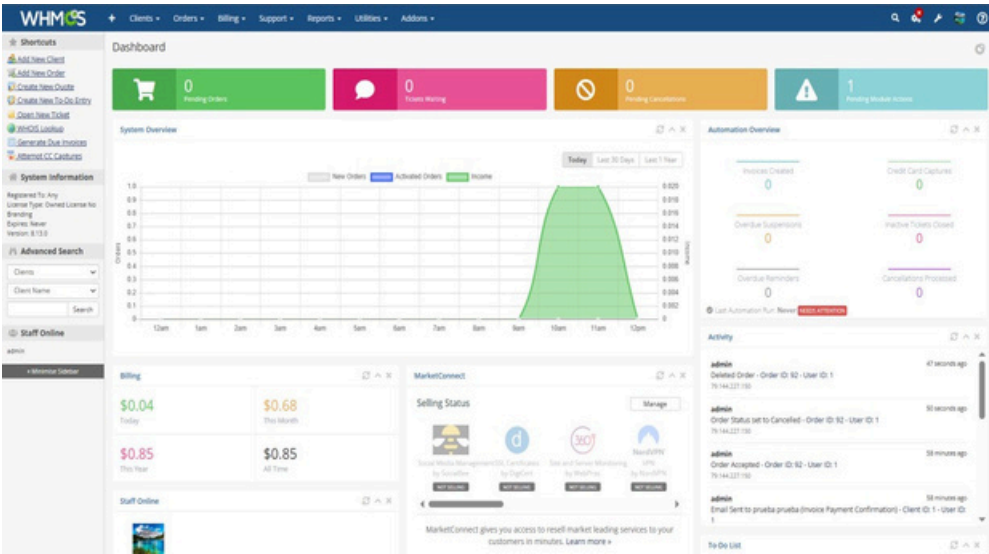
1. Interfaz Hestia CP:



2. Portal WHMCS: Vista desde cliente



3. Administración WHMCS:



4. Flujo n8n:

Webhook:

- Este nodo expone una URL pública que WHMCS llamará (cuando tú líneas el hook en includes/hooks/...) .
- En cuanto WHMCS marca la factura como pagada, envía un POST con datos básicos: serviceId, clientId, etc.
- n8n recibe ese JSON y lo pasa al siguiente nodo.

The screenshot shows the n8n Webhook node configuration. The 'Parameters' tab is active, showing a POST request to 'http://vpnsd.ddns.net:5678/webhook/whmcs-provision'. The 'HTTP Method' is set to 'POST', the 'Path' is 'whmcs-provision', and 'Authentication' is 'None'. The 'Respond' tab is set to 'Immediately'. The 'Options' tab shows 'No properties'. The 'OUTPUT' tab on the right displays a single JSON item:

```
[{"headers": {"host": "vpnsd.ddns.net:5678", "accept": "*/*", "content-type": "application/json", "content-length": "174"}, "params": {}, "query": {}, "body": {"invoiceId": 85, "clientId": 1, "serviceId": 93, "productId": 2, "clientEmail": "mmis5@punkproof.com", "ipAddress": "", "rootPassword": "", "hookToken": "d4F6gH8jK2LmN3pQ5rS7tU9vW0xY1zA2"}, "webhookUrl": "http://vpnsd.ddns.net:5678/webhook/whmcs-provision", "executionMode": "production"}]
```

Code JS: Se formatea la entrada JSON para extraer y normalizar campos necesarios (ServiceI, ClientId, productId etc).

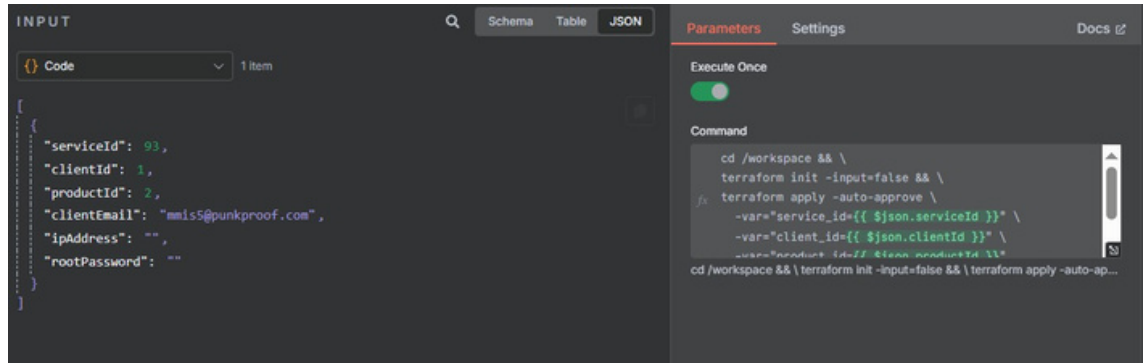
The screenshot shows the n8n Code node configuration. The 'Parameters' tab is active, showing the 'Mode' set to 'Run Once for All Items', the 'Language' set to 'JavaScript', and the 'JavaScript' code editor. The code is as follows:

```
1 // 1) Recuperamos el body completo
2 const p = items[0].json.body;
3
4 // 2) Validamos el hookToken
5 if (p.hookToken !== 'd4F6gH8jK2LmN3pQ5rS7tU9vW0xY1zA2') {
6   throw new Error('Token inválido');
7 }
8
9 // 3) Devolvemos solo lo que nos interesa
10 return [{
11   json: {
12     serviceId: p.serviceId,
13     clientId: p.clientId,
14     productId: p.productId,
15     clientEmail: p.clientEmail,
16     ipAddress: p.ipAddress,
17     rootPassword: p.rootPassword,
18   }
19 }];
```

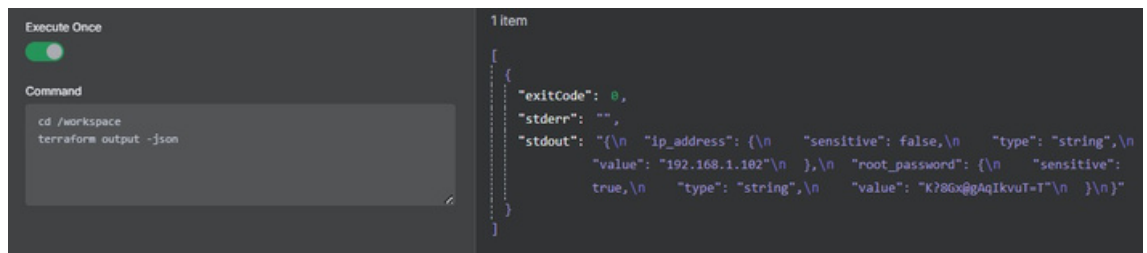
The 'OUTPUT' tab on the right displays a single JSON item:

```
[{"serviceId": 93, "clientId": 1, "productId": 2, "clientEmail": "mmis5@punkproof.com", "ipAddress": "", "rootPassword": ""}]
```

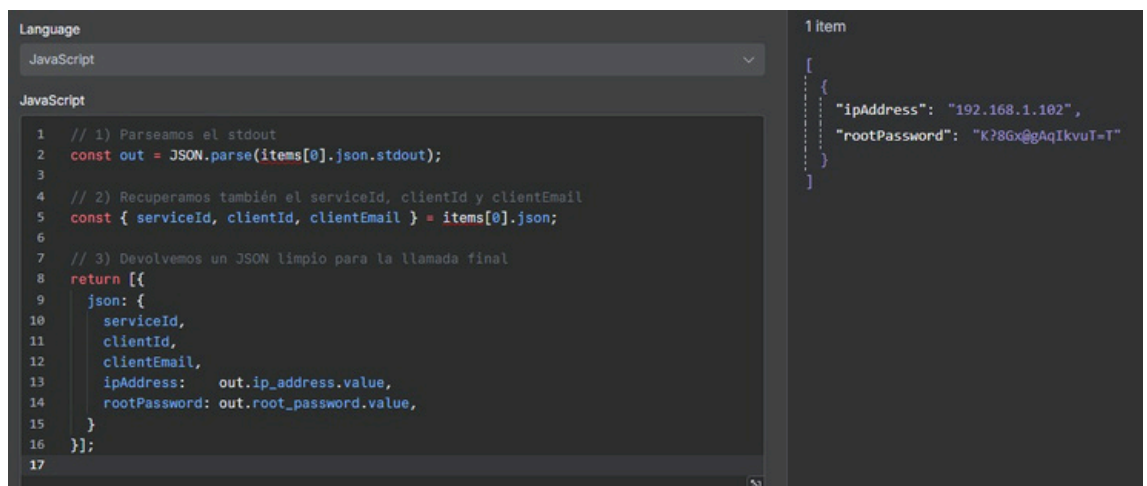
Execute Command: En este paso ejecutas el comando que realmente crea el VPS en tu proveedor



Execute Command1: Una vez creado el VPS, es habitual que haya que ejecutar otro comando para recuperar ip asignada y contraseña generada por cloud-init.

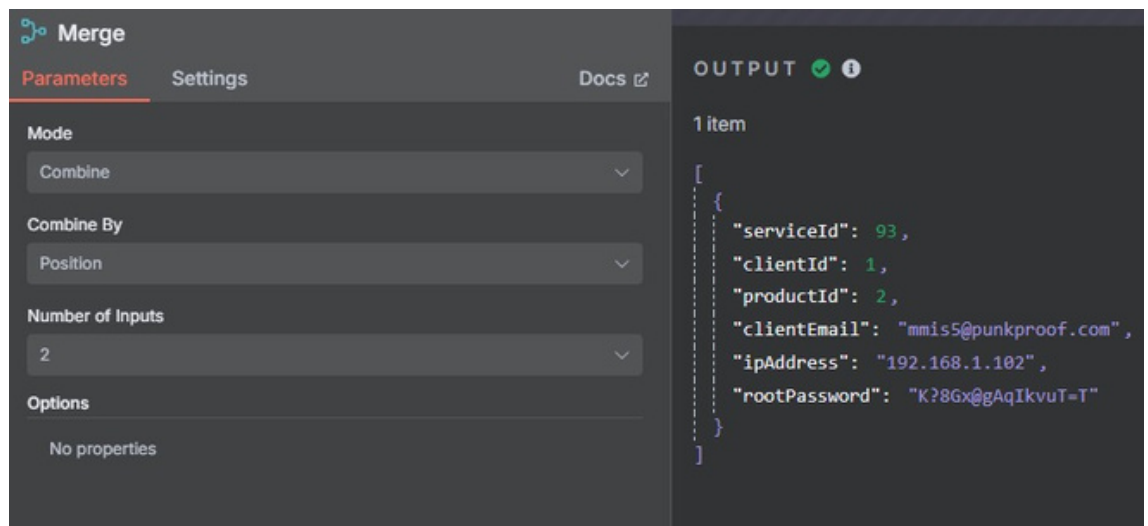


Code 1: Aquí tomas el resultado del paso 2 (datos de WHMCS) y del paso 4 (credenciales) y los combinas en un solo objeto JS,



Merge:

- Si por algún motivo llevas dos caminos paralelos —por ejemplo una rama que va directo al envío de email y otra que actualiza parámetros en WHMCS— con este nodo vuelves a juntarlos.
- Combinas “Input 1” (las credenciales + datos del cliente) con “Input 2” (credenciales del VPS) en un solo ítem.



The screenshot shows the configuration for a 'Merge' node. The 'Parameters' tab is active, showing the following settings:

- Mode:** Combine
- Combine By:** Position
- Number of Inputs:** 2
- Options:** No properties

The 'OUTPUT' tab shows the resulting JSON output for 1 item:

```
[
  {
    "serviceId": 93,
    "clientId": 1,
    "productId": 2,
    "clientEmail": "mmis5@punkproof.com",
    "ipAddress": "192.168.1.102",
    "rootPassword": "K?8Gx@gAqIkvuT=T"
  }
]
```

Code2: Darles el formato que la API de WHMCS espera

```
JavaScript
2 // Modo: Run Once for All Items
3 return items.map(item => {
4     // 1) Extraemos los valores de IP y rootPassword del JSON de entrada
5     const ip    = item.json.ipAddress    || "";
6     const pass  = item.json.rootPassword || "";
7
8     // 2) Construimos en JS un objeto con las claves que queremos serializar
9     //     Este objeto corresponderá al array PHP: ["ip_address" => ip,
10    "root_password" => pass]
11    const datos = {
12        ip_address: ip,
13        root_password: pass
14    };
15
16    // 3) Serializamos a formato PHP:
17    //     a:<número_de_elementos>:{s:<longitud_clave>:"clave";s:
18    <longitud_valor>:"valor"; ... }
19    let serialized = `a:${Object.keys(datos).length}:{`;
20    for (const [key, val] of Object.entries(datos)) {
21        // Buffer.byteLength(..., 'utf8') calcula la longitud en bytes (UTF-8) de la
22        // Buffer.byteLength(..., 'utf8') calcula la longitud en bytes (UTF-8) de la
23        // cadena
24        serialized += `s:${Buffer.byteLength(key, 'utf8')}:\"${key}\"`;
25        serialized += `s:${Buffer.byteLength(val, 'utf8')}:\"${val}\"`;
26    }
27    serialized += `}`;
28
29    // 4) Codificamos esa cadena serializada a Base64
30    const encoded = Buffer.from(serialized, 'utf8').toString('base64');
31
32    // 5) Guardamos la cadena Base64 en item.json.customvars
33    //     (será el valor que luego enviaremos en el HTTP Request)
34    item.json.customvars = encoded;
35
36    // 6) Devolvemos el item modificado
37    return item;
38 });
```

Resultado de Code2:

1 item

```
[
{
  "serviceId": 93,
  "clientId": 1,
  "productId": 2,
  "clientEmail": "mmis5@punkproof.com",
  "ipAddress": "192.168.1.102",
  "rootPassword": "K?8Gx@gAqIkvuT=T",
  "customvars": "YToyOntzOjEwOiJpcF9hZGRyZXNzIjtzOjEzOjE0IiwMTY4LjEu
    MTAYjtzOjEzOjEyb290X3Bhc3N3b3JkIjtzOjE2OjE0IjLPzhHeEBnQX
    FJa3Z1VD1UIjt9"
}
```

HTTP: Lanza a WHMCS el módulo VPS que está comprando el cliente y sale reflejado en los servicios que el cliente tiene comprados.

HTTP Request

Parameters Settings Docs

Name
action

Value
ModuleCreate

Name
responsetype

Value
json

Name
identifier

Value
v7s84k6yu7Yuwdfyrzh7FQgH1rdfmPDq

Name
secret

Value
f4wq45r05EhuZZ8jLNupdOCexqcwCmk4

Name
serviceid

HTTP Request

Parameters Settings Docs

Name
serviceid

Value
`fx {{json.serviceId}}`
60

Name
clientid

Value
`fx {{json.clientId}}`
1

Name
productid

Value
`fx {{json.productId}}`
2

Name
clientemail

Value
`fx {{json.clientEmail}}`
wgd0h@punkproof.com

HTTP Request

Parameters Settings Docs

2

Name
clientemail

Value
`fx {{json.clientEmail}}`
wgd0h@punkproof.com

Name
ipaddress

Value
`fx {{json.ipAddress}}`
192.168.1.102

Name
rootpassword

Value
`fx {{json.rootPassword}}`
K?8Gx@gAqlkvuT=T

Configuración Terraform

Main.tf

```
terraform {
  required_providers {
    proxmox = {
      source = "telmate/proxmox"
      version = "3.0.1-rc8"
    }
    random = {
      source = "hashicorp/random"
      version = "~> 3.0"
    }
  }
}

provider "proxmox" {
  pm_api_url      = "https://192.168.1.15:8006/api2/json"
  pm_api_token_id = var.proxmox_api_token_id
  pm_api_token_secret = var.proxmox_api_token_secret
  pm_tls_insecure = true
}

# Elige un índice aleatorio entre 0 y len(ip_pool)-1
resource "random_integer" "ip_index" {
  min = 0
  max = length(var.ip_pool) - 1
}

# Local con la IP elegida
locals {
  assigned_ip = var.ip_pool[random_integer.ip_index.result]
}

resource "random_password" "root" {
  length      = 16
  lower       = true
  upper       = true
  numeric     = true
  special     = true
  override_special = "!(@#%&*()-_+[]{}<>?)"
}

resource "proxmox_vm_qemu" "vps" {
  name       = var.hostname
  vmid       = var.vmid
  target_node = var.node
```

```
  clone      = var.template_name
  full_clone = true
  scsihw      = "virtio-scsi-single"

  # Disco principal
  disk {
    slot = "scsi0"
    storage = var.storage
    size = "20G"
  }

  # Disco Cloud-Init
  disk {
    slot = "ide2"
    type = "cloudinit"
    storage = var.storage
  }

  # CPU / RAM
  cores = 2
  memory = 2048

  # Red
  network {
    id = 0
    model = "virtio"
    bridge = "vbr0"
  }

  # Cloud-Init: inyecta usuario, pass e IP
  ciuser = "root"
  cipassword = random_password.root.result
  ipconfig0 = "ip=${local.assigned_ip}/24,gw=${var.gateway}"
}

output "ip_address" {
  description = "IP asignada"
  value       = local.assigned_ip
}

output "root_password" {
  description = "Contraseña root"
  value       = random_password.root.result
  sensitive   = true
}
```

Terraform.tfvars

```
proxmox_api_token_id      = "root@pam!terraform"
proxmox_api_token_secret = "b9ce7fd1-e884-4226-8890-4274496b7784"

vmid      = 201
hostname  = "vps-comprado"
#ip_address = "192.168.1.100"
template_name = "debian-clonar"
```

Variables.tf

```
variable "proxmox_api_token_id" {
  description = "ID del token API de Proxmox"
  type        = string
}

variable "proxmox_api_token_secret" {
  description = "Secreto del token API de Proxmox"
  type        = string
  sensitive   = true
}

variable "node" {
  description = "Nodo Proxmox donde crear la VM"
  type        = string
  default     = "pxe"
}

variable "template_name" {
  description = "Nombre de la plantilla en Proxmox"
  type        = string
  default     = "Debian-clonar"
}

variable "storage" {
  description = "Pool de storage para discos"
  type        = string
  default     = "local-lvm"
}

variable "vmid" {
  description = "VMID para la nueva VM"
  type        = number
}

variable "hostname" {
  description = "Hostname para la nueva VM"
  type        = string
}

variable "ip_pool" {
  description = "Lista de IPs a repartir"
  type        = list(string)
  default     = [
    "192.168.1.100",
    "192.168.1.101",
```

```
    "192.168.1.102",
    "192.168.1.103",
  ]
}

variable "gateway" {
  description = "Gateway de la red"
  type        = string
  default     = "192.168.1.1"
}

variable "service_id" {
  description = "ID del servicio en WHMCS"
  type        = string
}

variable "client_id" {
  description = "ID del cliente en WHMCS"
  type        = string
}

variable "product_id" {
  description = "ID del producto en WHMCS"
  type        = string
}

variable "config_options" {
  description = "Opciones configurables (JSON) de WHMCS"
  type        = map(any)
  default     = {}
}
```

Cloud-init que se puede integrar para creación de MV con parámetros personalizados

```
cloud-config
users:
  - name: jorge
    sudo: ALL=(ALL) NOPASSWD:ALL
    groups: users, admin
    home: /home/jorge
    shell: /bin/bash
    lock_passwd: false
    passwd: $6$/x81ivtpnFNysv.w$AitQoq0bFY21UNnLVdLa7eKmFZ/MTPt8DwxKqeivzyox1SuyR53zTQD1uGahN6j3rk1RB02403MVuOpe2rHJ30
```

Mantenimiento y Buenas Prácticas

- ☒ Backups: Automáticos de MySQL y `home/whmcs/mail`
 - ☒ Actualizaciones: Renovar certificados periódicamente.
 - ☒ Monitoreo: Integrar con Grafana/Prometheus.
 - ☒ Seguridad: Habilitar 2FA en n8n y WHMCS.
-

Arquitectura y diagrama

1. Descripción de la arquitectura

1. Cliente / Navegador

El cliente interactúa con la tienda WHMCS (checkout, registro, etc.).

2. WHMCS (Web + Hooks)

- Front-end de venta y gestión de servicios.
- Al completarse un pedido (AfterShoppingCartCheckout), ejecuta un hook PHP que:
 - i. Extrae `serviceId`, `clientId`, `productId`, `clientEmail` y añade el `hookToken`.
 - ii. Hace un POST a tu endpoint de n8n (`/webhook/whmcs-provision`), enviando el payload JSON.

3. n8n (Workflow Automation)

- Nodo Webhook: recibe el JSON de WHMCS.
- Nodo Code: valida `hookToken` y normaliza el payload.
- Nodo Execute Command (1): ejecuta `terraform init && terraform apply` con vars `serviceId`, `clientId`, `productId`, `clientEmail`.
- Nodo Execute Command (2): lanza `terraform output -json` y captura salida.
- Nodo Code (o Extract From File) + Edit Fields: parsea el JSON de Terraform y arma un objeto limpio con `ipAddress` y `rootPassword` más los IDs.
- Nodo HTTP Request: envía un POST `X-Www-Form-Urlencoded` a `/includes/api.php` de WHMCS, con:
 - `identifier`, `secret` (credenciales API),
 - `action=ModuleCreate`,
 - `serviceid`, `clientid`, `productid`, `clientemail`,
 - `ipaddress`, `rootpassword`,
 - `responsetype=json`.

4. Proxmox (Infraestructura)

Gestionado por Terraform, que provisiona la VM en tu cluster Proxmox y devuelve IP y credenciales.

5. API Interna WHMCS

- Recibe los datos de aprovisionamiento y termina el proceso, notificando al cliente y actualizando el estado del servicio.

IMÁGEN DE DIAGRAMA EXPLICADO MAS ARRIBA

```
%% Cliente
subgraph Cliente/Navegador Web
  A[Navegador Web]
end

%% WHMCS
subgraph WHMCS
  direction TB
  B[Checkout / Pedido completado]
  C[Hook PHP<br/>AfterShoppingCartCheckout]
  B --> C
  C --> D[n8n Webhook<br/>(/whmcs-provision)]
  F[API Interna WHMCS<br/>(includes/api.php)]
end

%% n8n Workflow
subgraph "n8n Workflow"
  direction TB
  D --> E1[1. Validar hookToken]
  E1 --> E2[2. Ejecutar Terraform Apply]
  E2 --> E3[3. Ejecutar Terraform Output -json]
  E3 --> E4[4. Parsear salida de Terraform]
  E4 --> E5[5. Mapear campos (ip, contraseña)]
  E5 --> G[6. HTTP Request<br/>ModuleCreate]
end

%% Infraestructura
subgraph "Infraestructura Proxmox"
  direction TB
  E2 --> T[Terraform sobre Proxmox]
  T --> E3
end

%% Flujos principales
A --> B
G --> F
```